上章回顾

- 重定向操作有哪几种?
- 如何自定义一个变量并赋值?
- 单引号、反撇号、双引号的作用是什么?
- 如何执行一个脚本?

本章内容

- 学会常见的条件测试操作
- 学会使用if条件语句

一、条件测试操作

1、使用test命令测试，条件成立返回 0 ， 不成立返回其他数值

2、共两种格式

1）test 条件表达式

2）[条件表达式]：条件表达式和中括号前后必须有空格 ******

3、文件测试实例

1）根据指定的路径判断是否为文件

2）也可以判断文件是否具有读、写、执行权限等

- **-d #是否为目录 directory**
- **-e #文件或目录是否存在 exist**
- **-f #是否是文件 file**
- **-r #是否具有读取权限 read**
- **-w #是否具有写入权限 write**
- **-x #是否具有执行权限 execute**

> 与运算符（&&）
>
> - 该运算符用于组合两个命令。只有当第一个命令成功执行（返回值为0）时，第二个命令才会被执行。
> - 例如：command1 && command2。如果command1成功执行，那么command2将被执行。
>
> 或运算符（||）：
>
> - 该运算符用于组合两个命令。如果第一个命令执行失败（返回值非0），则执行第二个命令。
> - 例如：command1 || command2。如果command1执行失败，那么command2将被执行。

```
[root@localhost ~] [ -d /media ]
[root@localhost ~] echo $?
0
[root@localhost ~] [ -d /media/cdrom ]
[root@localhost ~] echo $?
1
[root@localhost ~] [ -d /media/cdrom ] && echo "YES"
[root@localhost ~] [ -d /media ] && echo "YES"
```

```
[root@localhost ~] [ -d /media/cdrom ] && echo "YES" || echo "NO"
[root@localhost ~] [ -d /media ] && echo "YES"|| echo "NO"
```

4、**整数值比较**

- **-eq #等于 equal to**
- **-ne #不等于 not equal to**
- **-gt #大于 greater than**
- **-lt #小于 less than**
- **-le #小于或等于 less than or equal to**
- **-ge #大于或等于 greater than or equal to**

> 下列判断用户登录数量是否超出指定得值

```
[root@localhost ~] who | wc -l
1
[root@localhost ~] Unum=`who | wc -l`
[root@localhost ~] echo $Unum
1
[root@localhost ~] [ $Unum -gt 5 ] && echo "too many"
[root@localhost ~] [ $Unum -gt 5 ] && echo "too many" || echo "need more"
need more
```

> 下列判断系统内存是否可以满足使用

```
[root@localhost ~] free -m
              total        used        free      shared  buff/cache   available
Mem:            972         187         670           7         113         654
Swap:          2047           0        2047
[root@localhost ~] free -m | grep Mem:
Mem:            972         188         670           7         113         653
[root@localhost ~] free -m | grep Mem: | awk {'print $4'}
670
[root@localhost ~] FreeMem=$(free -m | grep Mem: | awk '{print $4}')
[root@localhost ~] [ $FreeMem -lt 1024 ] && echo ""$FreeMem"MB" || echo "内存足够"
663MB
[root@localhost ~] [ $FreeMem -lt 512 ] && echo ""$FreeMem"MB" || echo "内存足够"
```

```
内存足够
```

5、**字符串比较**

1）格式

- **[字符串1 = 字符串2]**
- **[字符串1 != 字符串2]**
- **[-z 字符串]**

2）常用的测试操作符

- **=：字符内容相同**
- **!=：字符内容不同**
- **-z：字符内容为空**

```
[root@localhost ~] echo $LANG
zh_CN.UTF-8
[root@localhost ~] [ $LANG != "en.US" ] && echo "NOT en.US"
NOT en.US
[root@localhost ~] read -p "是否覆盖现有文件（yes/no）？" ACK
是否覆盖现有文件（yes/no）？yes
[root@localhost ~] [ $ACK = "yes" ] && echo "覆盖"
覆盖
[root@localhost ~] read -p "是否覆盖现有文件（yes/no）？" ACK
是否覆盖现有文件（yes/no）？no
[root@localhost ~] [ $ACK = "no" ] && echo "不覆盖"
不覆盖
```

6、逻辑测试

1）&&，并且，必须每个命令都成立，使用test时&&可以替换为-a （与）

2）||，#或者，多条件中成立一条即可，使用test时||可以替换为-o （或）

3）！，不是 （非）

```
[root@localhost ~] uname -r
3.10.0-1160.el7.x86_64
[root@localhost ~] uname -r | awk '{print $1}'
3.10.0-1160.el7.x86_64

[root@localhost ~] uname -r | awk -F . '{print $1}'
3
[root@localhost ~] Mnum=$(uname -r | awk -F . '{print $1}')
```

```
[root@localhost ~] Snum=$(uname -r | awk -F . '{print $2}')

[root@localhost ~] [ $Mnum -eq 3 ] && [ $Snum -gt 5 ] && echo "符合要求"

[root@localhost ~] [ "abc" = "abc" ] && echo "真" || echo "假"

[root@localhost ~]  [  "abc" != "abc" ] && echo "真" || echo "假"

[root@localhost ~] [ "hello" = "sorry" ] && echo "真" || echo "假"

[root@localhost ~] [ ! "hello" = "sorry" ] && echo "真" || echo "假"

[root@localhost ~] [ $(id -u) -eq 0 ] && echo this is admin || echo this is not admin

```

7、编写一个测试用户是否存在的脚本

```
[root@localhost ~] vim ceshi.sh
#!/bin/bash
read -p "请输入需要判断的用户名: " user
id $user &> /dev/null
[ $? -eq 0 ] && echo "该$user存在" || echo "该$user不存在"
```

二、使用if条件语句

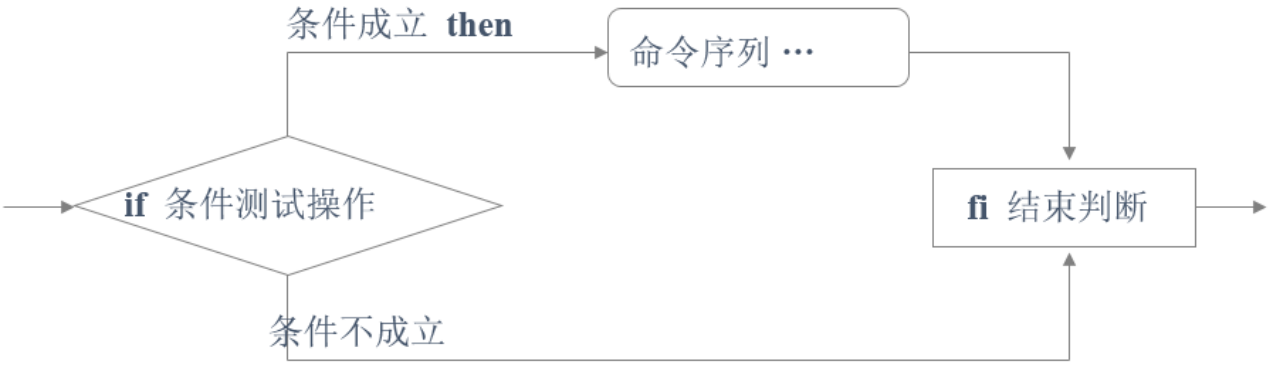
> if 语句是一种很常见的条件判断语句，被广泛应用在Python、C、C#、JAVA等各类语言中
> 基本功能就是根据判定的结果（真或假）给出两种或者多种操作之一

1、if 单分支结构

1）是一种简单的条件判断结构

2）基本语法为：if （条件表达式） {代码块}

3）简单来说就是判断条件是否成立，如果成立（True）则执行代码块中的代码；如果不成立（False）将会直接结束




```
if [ 条件判断式 ]; then
    命令1
fi
-----------
#!/bin/bash

if [ 1 -eq 1 ]; then
    echo "条件成立"
fi
```

2、if 双分支语句

1）与单分支不同的是可以有另一种选择

2）判断条件是否成立，分别指向不同的代码块




```
if 条件测试操作
    then   命令序列1
    else    命令序列2
fi
----------
#!/bin/bash
```

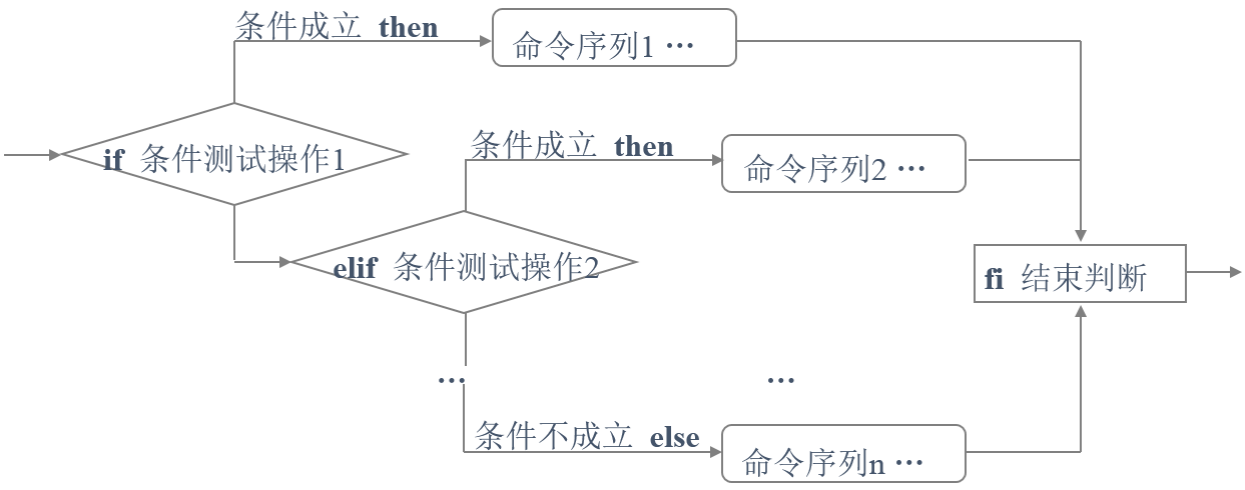
```
read -p "请输入分数:" GRADE
if [ $GRADE -ge 60 ];
        then
                echo "恭喜你！及格了！"
        else
                echo "没有及格，继续努力！"
fi


[root@localhost ceshi] source ceshi.sh
请输入分数:10
没有及格，继续努力!
[root@localhost ceshi] source ceshi.sh
请输入分数:65
恭喜你！及格了！
```

3、if 多分支语句

1）可以做N个判断条件，针对不同的条件输出不一样的结果

2）在多分支语法中可以存在多个elif




```
if 表达式1; then            ********
    # 如果表达式1为真，则执行这里的语句
elif 表达式2; then
    # 如果表达式1为假，但表达式2为真，则执行这里的语句
else
    # 如果所有表达式都为假，则执行这里的语句
fi
--------------
```

```
#!/bin/bash
read -p "请输入分数:" GRADE
if [ $GRADE -ge 80 ] && [ $GRADE -le 100 ]
        then
                echo "$GRADE分，优秀"
elif [ $GRADE -ge 60 ] && [ $GRADE -le 79 ]
        then
                echo "$GRADE分，及格"
else
                echo "$GRADE分，不及格"
fi
[root@localhost ceshi]# source ceshi.sh
请输入分数:99
99分，优秀
[root@localhost ceshi]# source ceshi.sh
请输入分数:78
78分，及格
[root@localhost ceshi]# source ceshi.sh
请输入分数:24
24分，不及格
```

三、if 分支语句案例

1、单分支案例

- 案例一：判断挂载点是否存在，如果不存在则创建挂载点目录

```
[root@localhost ~]# vim ceshi.sh
#!/bin/bash
MOUNT_DIR="/var/ftp/web"
if [ ! -e $MOUNT_DIR ]
    then
        mkdir $MOUNT_DIR
fi
```

- 案例二：判断当前用户是不是root

> 在shell中，"exit"命令是一个内置命令，它被用于退出当前的shell进程并返回一个退出状态。这个命令可以在终端或脚本中使用，语法为"exit [状态值]"，其中状态值0代表成功，其他值代表失败。
>
> 在脚本中，"exit"命令可以用于在任何时候终止脚本的执行，并且可以使用一个可选的退出代码来表示脚本终止的原因。例如，如果你希望在脚本执行到某个错误点时停止并返回一个特定的错误码，你可以使用"exit 1"（这里的1是退

> 出状态码，可以根据需要更改）来达到这个目的。

```
[root@localhost ~]# vim ceshi.sh
#!/bin/bash
if [ "$USER" != "root" ]
    then
        echo "错误：非root用户，权限不足！"
        exit 1
fi
fdisk -l /dev/sda
```

- 案例三：连通性测试脚本pinghost.sh

> -c 3 发送三个数据包
> -i 0.2 间隔0.2秒
> -W 3 超时3秒
> &> /dev/null 屏蔽ping命令执行过程的输出信息

```
[root@localhost ~] vim ceshi.sh
#!/bin/bash
ping -c 3 -i 0.2 -W 3 $1 &> /dev/null
if [ $? -eq 0 ]
        then
                echo "Host $1 is up"
        else
                echo "Host $1 is down"
fi
```

- 案例四：检查vsftpd服务是否运行

```
[root@localhost ~]# vim ceshi.sh
#!/bin/bash
systemctl status vsftpd &> /dev/null
if [ $? -eq 0 ]
        then
                echo "监听端口：$(netstat -anpt | grep vsftpd | awk -F: '{print $4}')"
                echo "进程PID号：$(pgrep vsftpd)"
                echo "vsftpd is running"
        else
```

```
                echo "警告：vsftpd服务不可用！"
fi
```

抄写作业

1、编写出if脚本的单分支、双分支、多分枝语法

2、如何测试/opt/backup目录是否存在?

3、如何测试是否已安装sysstat软件包

脚本作业

1、检测系统根目录下已使用空间，分支要求如下：

- 使用90%及提示空间不足
- 使用70% - 90%提示警告
- 使用50% - 70%提示用量
- 使用30% - 50%提示当前用量并且保持
- 使用30%一下提示空间充足6

2、结合计划任务，每2分钟执行一次，将结果输出到/data/name.txt文件中

- 输出的每一条包含自己的名字
- 并且在name.txt中输入时，每输出一次都有添加时间线和名字以及提示信息

> 命令提示

```
[root@localhost ~]# df -h
[root@localhost ~]# df -h |grep "/$"
/dev/mapper/centos-root 17G 5.5G 12G 32% /
[root@localhost ~]# df -h |grep "/$" | awk '{print $5}'
32%
[root@localhost ~]# df -h | grep "/$" | awk '{print $5}' | awk -F% '{print $1}'
32
[root@localhost ~]# CUG=$(df -h |grep "/$" | awk '{print $5}' | awk -F % '{print $1}')
[root@localhost ~]# date "+%Y-%m-%d %H:%M:%S"
2024-01-03 23:30:06

```

```bash
#!/bin/bash
current_time=`date "+%Y-%m-%d %H:%M:%S"`
CUG=$(df -h |grep "/$" | awk '{print $5}' | awk -F % '{print $1}')
if [ $CUG -gt 90 ]
        then
                echo "$current_time:已使用:$CUG%,空间严重不足"
elif [ $CUG -gt 70 ] && [ $CUG -le 90 ]
        then
                echo "$current_time:警告,已使用:$CUG%，空间即将不足"
elif [ $CUG -gt 50 ] && [ $CUG -le 70 ]
        then
                echo "$current_time:以使用:$CUG%，谨慎操作"

elif [ $CUG -gt 30 ] && [ $CUG -lt 50 ]
        then
                echo "$current_time:使用:$CUG%，继续保持"
else
        echo "$current_time:仅使用:$CUG%，空间充足"
fi

```